5

High Performance Protocol Architecture

W. Dabbous, C. Diot
INRIA Centre de Sophia Antipolis,
2004 Route des Lucioles, BP-93,
06902 Sophia Antipolis Cedex, FRANCE.
Tel: + 33 93 65 78 25, Fax: + 33 93 65 77 65
e-mail: {dabbous|cdiot}@sophia.inria.fr
http://www.inria.fr/rodeo/

Abstract

The performance enhancement of communication protocols is an essential step toward the development of high speed network applications. Application Level Framing (ALF) and Integrated Layer Processing (ILP) have been presented as two design principles for a new generation of protocols. In this paper, we study these high performance protocol design principles. We will first show the need for a new protocol architecture by presenting protocol optimization techniques and their limitations. We then describe ALF/ILP and study the impact of these principles on protocol design. Experiments with ALF and ILP are then presented. These experiments show that there is a performance gain when applying ALF/ILP to protocol design. We discuss the consequences of ALF and ILP based design on the way communication systems should be designed for more integration and more efficiency.

1 Introduction

The development of high speed networking applications requires efficient communication protocols. As networks proceed to higher speeds, the performance bottleneck is shifting from the bandwidth of the transmission media to the processing time necessary to execute higher layer protocols. In fact, the existing standard protocols were defined in the seventies. At that time, communication lines had low bandwidth and poor quality, and complex protocol functions were necessary to compensate for the transmission errors. The emergence of high speed networks has changed the situation, and these protocols would not be designed in the same way today.

At the same time, the application environment is changing. New applications (e.g. audio and video conferencing, collaborative work, supercomputer visualization etc.) with specific communication requirements are being considered. Depending on the application, these requirements may be one or more of the following: (1) high bit rates, (2) low jitter data transfer, (3) simultaneous exchange of multiple data streams with different "type of service" such as audio, video and textual data, (4) reliable multicast data transmission service, (5) low latency transfer for RPC based applications, etc.

The above requirements imply the necessity to revise the data communication services and protocols in order to fulfill the specific application needs. In fact, applications may "classically" choose either the connection-less or the connection oriented transport services. In both cases,

S. Fdida et al. (eds.), *Data Communications and their Performance*

the application needs are expressed in terms of quality of service (QoS) parameters such as e.g. the transit delay or the maximum throughput. However, the applications need more than a set of QoS parameters to control the transmission. The applications require to be involved in the choice of the control mechanisms and not only the parameters of a "standard" transport service. In fact, we argue that we need to define adaptation algorithms allowing a "Network Conscious Application" to govern the transmission. Placing most of the burden of network adaptation in the user equipment is in line with the "end to end argument", a key point of the Internet architecture. It contributes in keeping the network simple, and is very often the only way to scale up a complex system.

In this paper, we study and evaluate a new high performance protocol architecture. After a rapid survey on protocol optimization techniques in section 2, we discuss in some detail the need for a new protocol architecture and we present the ALF/ILP design principles in section 3. In section 4, we present experiments done in order to evaluate the performance gain obtained when the ALF/ILP principles are applied to the design of communication subsystems. The redesign of applications according to these principles may either be "manual" i.e. by applying the design rules directly resulting in a "network conscious application", or via a protocol compiler supporting theses design rules. Section 4.2 presents this "Network Conscious Applications" concept and section 4.3 shows how to perform automatic integration of communication systems in an efficient way according to the ALF/ILP principles. In section 5, we conclude the paper and present future work in this context.

2 High performance protocols

Early work on high performance protocols concentrated on the optimization of each layer separately. Concerning the transport protocols, several approaches have been studied such as the tuning of standard general purpose protocols [Wat87, Col85], the definition of new protocols or the work on enhanced implementations. In fact, good implementation techniques represent one of the most important factors in determining the performance of a given protocol [Cla89]. These techniques depend on the environment more than on the protocol itself. The proposed solutions focused on the enhancement of the protocol implementation performance in a given software or a hardware environment: outboard protocol processors (e.g. [Kan88], [Coo90]), hardware protocol implementations (early work on XTP/Protocol Engine [Ches89]) or parallel implementations of transport protocols [Brau92], [Rüt92], [Lap92], [Bjö93]. A detailed survey of protocol implementation optimization techniques can be found in [Dab91] and [Fel93].

Early work in the domain of enhanced transport *service* proposed the design of light weight special purpose protocols for specific application needs (e.g. NETBLT [Cla87a, Cla87b] for bulk data transfer and VMTP [Cher86] for transactional applications). This approach has a major limitation: the diversity of the applications increases the complexity of the transport by the support of several protocols. Each application would then choose a protocol corresponding to its specific needs. More recent research activities propose the synthesis of the so-called "communication subsystems" tailored to provide the service required by the application, from "building blocks" implementing elementary protocol functions such as flow control, error control, connection management (e.g. [Sch93], [Abb93b]). The synthesis of "fine grain" protocol functions should replace the "coarse grain" protocol choice (e.g. TCP or UDP).

At the presentation level, several research activities were centered around the optimization

of the ASN.1 Basic Encoding Rules. The cost of the coding and decoding routines is attributed to the heavy Type-Length-Value oriented coding of ASN.1 BER. This motivated the work on "light weight" or XDR-like transfer syntaxes (LWS) [Hui90, Hui89] based on three design principles: (1) avoid unnecessary information in the encoding, (2) use fixed representation when it is possible, and (3) simplify the mapping of the elements by fixed length structures. In parallel, the optimization of the BER implementation functions was conducted and resulted in a drastic improvement of the speed of coding and decoding routines on high performance RISC workstations. We tested an enhanced version of the MAVROS compiler [Hui91] with improved performance for the generated coding and decoding routines [Dab92]. In table 1, we present the results of a performance comparison test for coding and decoding routines of a complex data type (X.400 message) on several workstations.

	Coding time (μs)		Decoding time (μs)		Size (octets)		Decoding (Mbps)	
	BER	LWS	BER	LWS	BER	LWS	BER	LWS
Sun3	1861	2018	6944	6398	540	1029	0.62	1.29
SS 10/30	112	199	318	225	540	1029	13.58	36.59
Dec 5000/300	208	208	372	212	540	1029	11.6	38.8
HP 9000/715	100	100	339	155	540	1029	12.74	53.1
Dec alpha	74	55	176	92.6	540	1029	24.54	88.9

Table 1: Speed of the presentation routines

These results show that on high performance RISC workstations such as SparcStation 10, coding BER is much more efficient than LWS, which implies that the memory access dominates the processing time. However, BER decoding is still limited by the CPU performance. Both coding and decoding routines for LWS are limited by the memory access (due to the large code size stored and loaded). Similar results are obtained on both Dec 3000 and HP workstations. Note that the coding time is the same for both BER and LWS, which means that both memory access and CPU bottlenecks are balanced. For decoding, the processor speed limitation is more pronounced. The "best" results are obtained on Dec-alpha where the 64-bit bus enhances the memory access performance. The figures are "classical": BER is more costly than LWS, and decoding (BER or LWS) is more costly than coding. This is typically due to CPU speed limitations.

From the above results we can learn the following:

- A drastic improvement of the speed of coding and decoding routines for both BER and LWS routines can be obtained by adequate implementation optimization on high performance workstations
- The LWS is interesting if the processor speed is the bottleneck; however, a more compact syntax is desirable in order to reduce the size of the data to be transmitted on the network and the memory bus.
- Memory access is a limiting factor in most cases on RISC workstations. This confirms that integration techniques should result in increased performance on such workstations.

The optimized version of the encoding and decoding functions should facilitate the implementation of the presentation layer as a filter with "streamlined" *encoding* and *transmission* of application data units.

All the activities cited above in this section were focused on the optimization of specific "layers" independently. We argue that the integration of all the application communication requirements (including functions corresponding to the OSI transport, session and presentation layers) in order to generate a single protocol automaton for the application will result in increased performance gain. In this case, the application selects the options that govern the data exchange, i.e., the policy used for the transmission control. In fact, only the application has sufficient knowledge about its own requirements to optimize the parameters of a data exchange in a convenient way. The application can easily adapt to the network resources changes by appropriate steps: instead of reducing the window size at the transport level in response to a congestion indication, a video-conference application may chose to degrade either the quality[1] or the frequency of the images in order to adapt to the available bandwidth.

This approach is not consistent with the layered OSI model, where clear separation of data transmission control mechanisms (lower layers) and data processing functions (higher layer) is made. The OSI approach discharges the application from the transmission control functions by defining the transport service. The applications are *transport service users* according to the layered reference model. On the contrary, the integrated design and implementation approach put the application inside the "control loop", but it still needs to define how the application level parameters will be mapped onto network parameters and control functions. According to the diversity of the applications two solutions are possible:

- either we define application classes and we select appropriate control functions to be integrated into each class.
- or we use a suitable specification language to express the applications needs and we design a generic tool to derive the complete communication subsystem automatically.

The first solution means an extension of the transport service. There has been considerable work on the definition of high speed transport service(s) [Léo92],[Diot92]. However, this solution seems limited due to variety of application profiles. A more focused solution is desirable.

The second approach is to build specialized communication subsystems based on specific application requirements and on a set of optimized building blocks. This corresponds to a horizontal approach to the layered architecture, i.e., applications select control and data manipulation functions based on the service parameters and on the cost of these functions. These functions should be combined in a way that minimizes the memory access cost, in order to build the complete communication subsystem.

These ideas are in line with the ALF/ILP design concepts proposed by Clark and Tennenhouse [Cla90]. The next section will be dedicated to the presentation of these concepts.

[1] The quality of a H.261 image can be controlled by changing he quantizer value and/or the movement detection threshold. These two parameters are used in the INRIA Videoconference System' control algorithm to adapt the quality of the image to the available bandwidth.

3 New protocol architecture

The requirement for a new protocol architecture is clear. Traditional layered protocol architectures such as the ISO OSI model (see for example [IEEE83]) and the ARPA Internet model (see for example [Lei85]), are reaching the very end of their lifetimes.

The artificial separation of concerns in layers is partly due to the hardware architecture of the 70's. The service/protocol concept derives historically from the model presenting interfaces between different service providers. There would be a link, network, transport and session provider, perhaps all of which would have different potential vendors. This approach has proved one thing: The inefficiency of the communication systems pushed buyers to go to other markets. The market appears to be in 3 layers: end system hardware and operating systems; end systems communications stacks; finally transmission networks.

Even in a single vendor software implementation, one could accuse the layered model (TCP/IP or OSI) of causing inefficiency. In fact, the operations of multiplexing and segmentation both hide vital information that lower layers need to optimize their performance. Although it is important to distinguish between the architecture of a protocol suite and the implementation of a specific end system or a relay node, the layered protocol architecture may unnecessarily reduce the engineering alternatives available to an implementor. In fact, a certain degree of flexibility in the way the functions are organized within the layers is needed. A proposed solution is to integrate all transmission control layers in a single block, in essence discarding the highly modular layered model in exchange for performance.

3.1 ALF/ILP

In their SIGCOMM '90 paper [Cla90], Clark and Tennenhouse have proposed Application Level Framing (ALF) as a key architectural principle for the design of a new generation of protocols. ALF is in fact the natural result of advanced networking experiments, which showed the need for "a rule of three units:"

1. a relatively old result is that efficient transmission can only be achieved if the unit of control is exactly equal to the unit of transmission. Obvious penalties for violating this rule are unnecessary large retransmissions in case of errors and inefficient memory usage.

2. the key idea expressed in [Cla90] is that the unit of transmission should also be the unit of processing. Otherwise, large queues will build up in front of the receiving processes and eventually slow down the application.

3. we found, through experience with multimedia services, that adaptive applications are much easier to develop if the unit of processing is also the unit of control.

According to the ALF principle, applications send their data in autonomous "frames" (or Application Data Units (ADUs)) meaningful to the application. It is also desirable that the presentation and transport layers preserve the frame boundaries as they process the data. In fact, this is in line with the widespread view that multiplexing of application data streams should only be done once in the protocol suite. The sending and receiving application should define what data goes in an ADU so that the ADUs can be processed out of order. The ADU will be considered as the unit of "data manipulation", which will simplify the processing. For example, an image server will send messages corresponding to well identified parts of the picture. When a receiver

receives such a frame, it can immediately decompress the data and "paint" the corresponding pixels on the screen, thus minimizing response times.

Integrated Layer Processing (ILP) is an engineering principle that has been suggested for addressing the increased cost of data manipulation functions on modern workstations. In fact, the performance of workstations has increased with the advent of modern RISC architectures but not at the same pace as the network bandwidth during past years. Furthermore, access to primary memory is relatively costly compared to cache and registers, and the discrepancy between the processor and memory performance is expected to get worse. The memory access is expected to represent a bottleneck [Dru93].

Protocol processing can be divided into two parts, control functions and data manipulation functions. Example of data manipulation functions are presentation encoding, checksumming, encryption and compression. In the control part there are functions for header and connection state processing. Jacobson et al. have demonstrated that the control part processing can match gigabit network performance for the most common size of PDUs with appropriate implementations [Cla89].

Data manipulation functions present a bottleneck [Cla90], [Gun91]. They consist of two or three phases. First a read phase where data is loaded from memory to cache or registers, then a manipulation or "processing" phase, followed by a write phase for some functions, e.g., presentation encoding. For very *simple functions*, e.g. checksumming or byte swap, the time to read and write to memory dominates the processing time. For other *processing oriented functions*, like encryption and some presentation encodings, the manipulation time dominates with current processor speeds. However, the situation is expected to change with the increase of processor performance: the memory access will be the major bottleneck rather than data processing.

The data manipulation functions are spread over different layers. In a naive protocol suite implementation, the layers are mapped into distinct software or hardware entities which can be seen as atomic entities. The functions of each layer are carried out completely before the protocol data unit is passed to the next layer. This means that the optimization of each layer has to be done separately. Such ordering constraints are in conflict with efficient implementation of data manipulation functions [Wak92], [Cla90].

The main concept behind ILP is to minimize costly memory read/write operations by combining data manipulation oriented functions within one or two processing loops instead of performing them serially as is most often done today . It is expected that the cost reduction due to this optimization will result in better overall performance as it will reduce time consuming memory access. This optimization may be applied within a single data manipulation function (*intra-function optimization*) or across several functions (*inter-function optimization*) [Dab94a]. The interest of the ILP principle (and similar software pipelining principles such as lazy message evaluation and delayed evaluation) has been discussed in [Cla90], [Gun91], [Par93b], [O'M90], [Peh92] and [Abb93a].

Previous work ([Gun91] [Dab94a] [Par93b]) has demonstrated that there is a performance benefit with ILP. The results show reduced processing time for one PDU, as much as a factor of five when six simple data manipulation functions were integrated. These reported results came from experiments that were isolated from the rest of the protocol stacks and where hand-coded assembler routines were used in order to control register allocation and cache behavior. Abbot and Peterson [Abb93b] use a language approach to integrate functions, but with less speedup. In [Par93b] only two functions are integrated, data copying and checksum calculation,

but it is a real operational implementation of UDP. Experience with an implementation of the XTP protocol from the OSI95 project [Dab93] showed improved performance when ILP was used. This implementation is in user address space, which also demonstrates that such implementations can perform as well as kernel tuned implementations. We should go one step further and integrate several of the data manipulation functions in a complete, operational stack in order to understand the architectural implications and the achievable speedup.

ALF and ILP were proposed in 1990 and for the time being there is no complete implementation of a communication subsystem based on both concepts. Several reasons make the design of a global framework for integrated implementation based on ALF/ILP concepts now both feasible and attractive:

- The increase of processor speeds and of the discrepancy between processor and memory performance is pushing toward the integration of data manipulation operations.
- Experience with an implementation of the XTP [Dab93] and TCP [Hogl94, Cas94a] protocols showed that user level software implementations can perform as well as kernel tuned TCP, while still keeping the flexibility of configurable software.
- Performance enhancements to the ASN.1 encoding facilitate the implementation of a presentation filter [Dab92].
- Multiple communication services including group communication and variable error control have been studied for a variety of applications (e.g. multimedia video conferences with shared workspace, mobile applications). It is very desirable to have a mechanism to customize communication subsystems to specific application needs.

4 Impact on protocol design

The basic idea of ALF and ILP is that the protocol should adapt to the requirements of the application. The application is acknowledged to be best prepared to choose the proper strategies for treating lost or out-of-order data. This design of a flexible integrated architecture requires on the one hand that the communication system have access to the application semantics, and on the other hand that the application have the means to affect the relevant control and synchronization aspects of the communication system.

In order to validate this integrated approach, we wanted to analyze and measure what the real impact of ALF and ILP was on both the architecture and the performance of the communication system.

This section presents first the experiments we led with ILP and ALF. The most significant results are shown. Then, we discuss the consequences of ALF and ILP based design on the way communication systems should be designed for more integration and more efficiency.

4.1 Experimental results

- ALF evaluation. Two hand-coded implementations of a JPEG player have been designed to investigate the effect of ALF on communication subsystems design and performance. The No ALF implementation runs over an in-kernel TCP. The ALF implementation runs over its own protocol which takes advantage of ALF. The protocol called TPALF is a

user-level protocol that runs over UDP/IP. It was modified from the 4.3BSD TCP/IP implementation. The only changes with regard to TCP were to allow out-of-order processing of incoming data and to handle ADUs as opposed to streams of packets. Flow control is done with a sliding-window scheme using the slow-start algorithm. Error control is achieved through both Cumulative and Selective Negative Acknowledgments.

A first issue was to determine the size of the unit of transmission. As JPEG ADUs are small, several ADUs have been concatenated within one *transmission unit* or NDU (Network Data Unit). But ALF means also the ADUs must be preserved through the whole communication system and NDU segmentation must be avoided. ALF is a strategy that organizes the transmitted NDUs into data meaningful to the application. This allows the receiver to process independently and immediately each packet received. When segmentation occurs within the protocol, the received packets cannot be delivered to the application on arrival, and the benefits of ALF are lost (table 2). Thus the size of the NDU should not exceed the size of the minimum MTU (Maximum Transmission Unit) of the network.

MTU	ALF	No ALF (TCP)
512 bytes	14.7 Kbits/s	6.3 Kbits/s

Table 2: Throughput via Internet, NDU size = 512 octets

MTU	ALF	No ALF (TCP)
1460 bytes	7.35 Mbits/s	7.67 Mbits/s

Table 3: Throughput via local Ethernet, NDU size = 1460 octets

The two hand-coded implementations have been also used to investigate whether applications can benefits from out-of-order processing. The comparison between table 2 and table 3 demonstrates how non-ordering gives the possibility of exploiting the internal parallelism of the application. The experiments through local networks (table 3) show that the No ALF implementation[2] performs slightly better when the underlying network is reliable (almost no out-of-sequence data transmission). Through Internet (table 2), where delays and loss can produce out-of-order data delivery, ALF appears to be more efficient (more than 200 %) because the receiver is able to process the ADUs immediately when they arrive, whether they are in order or not. ALF tends to improve the efficiency of the communication sub-system by handling the ordering at the application level. Further details are given in [Diot95].

- Integrated Layer Processing is an implementation concept "to permit the implementor the option of performing all the manipulation steps in one or two integrated processing loops" [Cla90]. Integration of data manipulations by ILP seems to be a promising strategy to increase the performance of communication systems. During the last years the

[2]In the no ALF case, the communications are done over a reliable TCP stream.

performance of processors has been increasing very fast compared to the performance of memories. The memory bottleneck can be reduced by simply avoiding the access to the memory as much as possible, e.g. by eliminating copy operations from the protocol implementations. If avoiding memory access is not further applicable, another solution is to use caches, which are extremely fast but expensive. Usually the fast caches on the processor chips have limited sizes in the range of a few Kbytes (16 Kbytes data cache and 20 Kbytes instruction cache on a SuperSPARC10 processor). To get benefits from the cache it is necessary to keep as much data as possible within the cache.

The concept of ILP tries to gain from both approaches. Theoretically, an ILP protocol stack implementation reads once from the main memory, keeps the read data within registers or cache memory, and performs all the data manipulations of several protocol layers. Processed data are directly written to the destination memory. In this ideal case, the ILP approach requires only one read and one write access to the main memory for each word. All the other operations should work on registers, and eventually on the cache memory. This integrated processing of data is commonly called the ILP loop.

By applying the ILP technique on simple data manipulations like data copying and TCP checksum calculation, performance gains of 50% have been achieved [Ten89]. A similar but also simple experiment nearly shows the same results. The XDR marshalling routine obtained from a stub compiler for an array of 20 integer values has been combined and executed with the TCP checksum routine. When executing the two routines sequentially, the throughput was 70 Mbps, in contrast with the 100 Mbps observed when integrating both functions into a single loop. The performance comparison shows 30% gain in favor of the ILP implementation. The advantages by integrating more data manipulation functions may lead to even larger benefits.

Other simple experiments showed that ILP is sensitive to data manipulation functions. The integration of the DES algorithm (data encryption standard) can reduce the performance gains significantly [Gun91]. But benefits that can be obtained from ILP not only depend on the data manipulation function complexity, but also on other characteristics of these functions like the number and the size of required memory tables, and the necessary interaction between data manipulations and control functions. This means that experiments performed only on data manipulation functions cannot show the real advantages of an ILP implementation.

Designing an experimental ILP implementation of a three level protocol suite based on a user-level TCP, we discovered that ILP reduces the number of memory accesses by 26%, but the relative amount of cache misses could never be reduced compared with a carefully designed non-ILP implementation [Brau95]. ILP throughput improvements are limited and depend heavily on several issues such as complexity of data manipulations, communication subsystem architecture, and host environment characteristics. In the experiment, these reasons decrease the throughput gain to approximately 10% in contrast to the 40 to 50% gain achieved for simple loop experiments. It is shown that ILP is very sensitive to several issues. That makes its use in existing communication systems and workstations debatable.

ILP has the main limitation that it is only applicable with certain types of protocol functions (non-ordering constraint functions) and protocol architectures (header size must be known before data manipulation processing). Another major drawback of ILP is the re-

duced flexibility, because the use of macros instead of function calls is required to avoid performance loss. Macros do not allow a protocol implementation to be adapted dynamically to changing application requirements or to varying network characteristics.

These experiments with ALF and ILP show that efficient implementation of distributed multimedia applications requires new architectural considerations and not only "enhanced protocol implementation techniques". However, using advanced protocol features such as fixed size headers, different packet types for control information and data, uniform processing unit sizes for different data manipulation functions could also be advantageous for ILP and ALF.

4.2 Network Conscious Applications

ALF implies that applications control their transmission. This may seem an undesirable feature. In fact, ATM is going to provide us with very high bandwidth, that applications should just aim for the best and ask for the corresponding resources. We argue however, that, on the contrary, applications should be designed with "inter-networking" in mind and they should absolutely include the proper adaptation loops. We discuss in this section, the implication of this rule on applications design. Clearly, networks are getting more and more faster. But as the technology progresses, the networking conditions are going to become more and more variable. We will have T1 and T3 lines, 10baseT and 100baseT networks, 640 Mbps Myrinet graphs, 155 Mbps ATM circuits and very variable capacities mobile networks. It should be quite obvious that the applications which can gracefully adapt to multiple environments will outlive those which have to die when their requirements are not met.

It should also be quite obvious that an application which can automatically characterize its environment will be more robust and easier to deploy than an application which has to exchange signalling information with the network for that purpose. Building the characterization loop and the adaptation code inside the application does indeed not prohibit reservation of resources. It simply decouples it from the application design. Adaptability guarantees that the application will always use all the available resources and make the most efficient usage of these resources. That these resources result from a reservation or simply from a best effort network is entirely irrelevant. On the other hand, communication over a network is not easy to guarantee. Communication protocols have been designed to minimize the unreliability of transmission over a net work. Therefore the application adaptivity is always a desirable feature because:

- Even if resource reservation is done, it will be done with a certain level of reliability, specifying a minimum and a desirable (or average) throughput. The adaptation of the application throughput can help guaranteeing the best quality of transmission within the throughput interval defined by QoS negotiation, and optimizing network resource utilization.

- Priorities can be associated to applications (following the level of service required, or the cost payed for the service). If higher level of priority reservation occurs, resource will be taken from lower priority communications. Guarantee will only be effective for higher level priority applications.

- Routers with guaranteed resource may crash. Re-routing on another router, using a new path, does not guarantee that the same resource will be available.

This concept implies that the same application which usually use high data rates may occasionally be constrained to use a few tens of kilobits. The application should then be designed to pro-actively select the most important data for transmission when the resource is scarce, rather than merely accept to be slowed down or to let the network randomly drop some messages.

We argue that this adaptation concept is very useful even for time-constrained applications (i.e. multimedia, voice, video). The devotees of "QoS models" will oppose this argument; they would rather prefer a transmission to be stopped than to see their Quality of Service not respected. Imagine being at home, watching a TV controlled by a QoS manager. You wouldn't like your TV to be switched off if a temporary degradation of the QoS occurs. Stop a communication on QoS degradation is a nice theoretical behavior; but in most of the multimedia applications, temporary and controlled graceful degradation of quality will be preferred to the connection being released.

We have experimented the "Network Consciousness" concept on IVS [Turl94b], a software system to transmit audio and video data over the Internet. It includes PCM and ADPCM audio codecs, as well as a H.261 video codec. In the initial versions of IVS, the user could control manually the maximal output rate of the coder. The variable video output rate was sent into a buffer which was drained at a constant rate. Then, the amount of data in the buffer was used as a feedback information to adapt the parameters of the coder in order not to exceed the maximal output rate allowed. The output rate was controlled by changing either the video frame rate or the quantizer value and the movement detection threshold.

In order to behave like a "good network citizen", we improved the application by adding a feedback control mechanism that prevents the video application from swamping the resources of the Internet. In this mechanism, the parameters of the coder were adjusted according to the network conditions [Bol94a]. Receivers periodically sent back to the coder the observed loss rate and the coder computed an average loss rate to estimate the network load and selected a maximal output rate value according to it.

However, the previous scheme did not scale up to a very large number of receivers: in this case a large amount of feedback is sent to the coder which would cause feedback implosion problem. To scale up to any number of receivers, we implemented a scalable feedback mechanism based on a novel probing mechanism to solicit feedback information in a scalable manner [Bol94b]. The feedback information also includes a loss rate indication sent back from the receivers to the coder. Given this feedback information, the number of receivers with low video quality reception is estimated and the video coder uses it to decide how to adjust its maximal output rate.

This example shows that is both possible and desirable to implement adaptive applications, as implied by the ALF principle.

4.3 Automatic integration

The traditional empirical methodology for protocol design and implementation, which is mostly "intuition based on experience", will not be able to scale up effectively and to produce highly integrated implementations for the new generation of applications. To be able to design efficient implementations of communication support tailored to application characteristics, we claim that the development process of communication support must largely be automated (in a formal framework). This will allow to "easily" generate a large panoply of communication subsystems tailored to application needs. A formal automated design process also allows the correctness of the communication to be checked both in terms of reliability and security, and even the efficiency

can be determined given a set of constraints.

The Protocol Compiler is a step toward the support of a new generation communication model, where a distributed application can specify its own communication requirements to be associated to a dedicated transmission control protocol. The control and synchronization aspects of an application are formally specified using ESTEREL [Berr92, Cas94b]. This specification is then used by the protocol compiler to integrate transmission control facilities to the application, and to generate the client and server stubs. The advantages of this approach are the following:

- Like in the classical communication model, the transmission facilities remain transparent to the application designer.
- The control of the transmission is given to the application. The communication facilities are integrated in the application. This allows the application to adapt dynamically its throughput to the available resource on the network.
- The implementation is designed automatically, starting from the application's formal specification. It guarantees that the implementation matches exactly with the original application specification. It is also easier to prove that the communication system provides the application with minimal and efficient communication support.
- The formal approach allows the systematic use of optimization techniques such as optimizing the protocol control automaton, discovering the most frequently used path, in-lining code, etc. These optimizations might lead to better performance than with hand-coded implementations.

Other research groups are currently working on the design and implementation of communication subsystems tailored to application requirements [Oech94], [Plag94], [Sch93], [Rich94], [Diaz94], [Omal94]. The proposed solutions include developing general purpose protocols that allow flexibility. However, these solutions are not operating system independent because the implementations are either part of the kernel, or a server within a micro-kernel based operating system.

The approach that we propose is different, as it starts from the application specification, and as the transmission control facilities are integrated to the application specification before the automated implementation. Flexibility is allowed by the integration of the transmission control mechanism within the application.

A prototype Protocol Compiler has been designed [Diot95]. The JPEG player, hand-coded for the evaluation of the ALF concept, has also been implemented automatically starting from its ESTEREL specifications, and using the Protocol Compiler. With similar protocols, the automated implementation runs 20% faster than the hand-coded on experiments between INRIA (France) and UTS (Australia) (see table 4).

On the code size aspect, the Protocol Compiler produces a code which has quite the same size as the hand-code written in C language. The executable code of the receiver side is even smaller (4% on 200 Kbytes). Recent optimization work on the ESTEREL's code generation phase show that a sensible reduction of the generated code size is possible without a loss of performance. This is very promising for our automated approach.

	Local Ethernet	Internet
handcoded ALF	7.39 Mbits/s	51.3 Kbits/s
ESTEREL	7.42 Mbits/s	62.1 Kbits/s

Table 4: Throughput of both hand-coded and ESTEREL versions

5 Conclusion

Based on the evaluation of ILP and ALF concepts, this paper shows that a new high performance protocol architecture is desirable for efficient implementation of multimedia applications. We propose to base this efficient protocol architecture on three concepts:

- ALF, which is a design principle allowing efficient processing of the application data units;
- Network conscious applications will be able to adapt to resources available on the network, even in association with guaranteed bandwidth services.
- Automated integration of transmission control facilities tailored to the application requirements. We showed its feasibility in the case of an experiment with a JPEG image player.

These results are being applied with the HIPPARCH project to the design of a new generation Protocol Compiler dedicated to multimedia applications. It has been demonstrated that the automated integration of transmission control functions in an application formal specification is possible in practice. Performance results confirm that, in term of code organization, size, and efficiency, the automated approach is almost as efficient as the hand-coded approach. Using this approach, a completely automated and efficient implementation of distributed applications is possible.

Acknowledgments

The authors would like to thank Isabelle Chrisment who developped the JPEG server for providing the results of the tests.

References

[Abb93a] Mark B. Abbott, Larry L. Peterson. "Increasing Network Throughput by Integrating Protocol Layers ", *IEEE/ACM Transactions on Networking*, Vol 1, No 4, October 1993.

[Abb93b] Mark B. Abbott, Larry L. Peterson. "A Language-Based Approach to Protocol Implementation", *IEEE/ACM Transactions on Networking*, Vol 1, No 1, February 1993.

[Berr92] G. Berry, G. Gonthier. "The Esterel Synchronous Programming Language: Design, Semantics, Implementation". *Journal of Science Of Computer Programming*, Vol. 19, Num. 2, pp. 87-152. 1992.

[Björ93] M. Björkman and P. Gunningberg. "Locking Effects in Multiprocessor Implementation of Protocols", In *Proceedings ACM SIGCOMM'93.*

[Bol94a] J.-C. Bolot, T. Turletti. "A rate control mechanism for packet video in the internet," *in Proceedings of the Conference on Computer Communications, IEEE Infocom '94*, Toronto, Canada, June 1994.

[Bol94b] J-C. Bolot, T. Turletti, I. Wakeman. "Scalable feedback control for multicast video distribution in the Internet", Proc. ACM SIGCOMM '94, Vol. 24, No 4, October 1994, pp. 58-67.

[Brau92] T. Braun and M. Zitterbart. "Parallel Transport System Design", In *Proceedings of the* 4^{th} *IFIP Conference on High Performance Networking*, Liège, Belgium, December 1992.

[Brau95] T. Braun and C. Diot. "Protocol Implementation using ILP", SIGCOMM '95, Boston, August 1995.

[Cas94a] Claude Castelluccia and Walid Dabbous. "Modular Communication Subsystem Implementation using a Synchronous approach", In *Proceedings of USENIX-94, Symposium on High Speed Networking*, Oakland, CA, August 1994.

[Cas94b] C. Castelluccia, I. Chrisment, W. Dabbous, C. Diot, C. Huitema, E. Siegel. "Tailored Protocol Development Using ESTEREL," INRIA Research report, No 2374, October 1994.

[Cher86] D. R. Cheriton, "VMTP: a transport protocol for the next generation of communication systems", In *Proceedings ACM SIGCOMM '86,* Stowe, Vermont, August 1986, pp. 406-415.

[Ches89] G. Chesson, "XTP/PE Design Considerations", In *Protocols for High-Speed Networks,* H. Rudin, R. Williamson, Eds., Elsevier Science Publishers/North-Holland, May 1989.

[Cla90] David D. Clark and David L. Tennenhouse. "Architectural Considerations for a New Generation of Protocols", In *Proceedings ACM SIGCOMM '90,* September 24-27, 1990, Philadelphia, Pennsylvania, pp. 200-208.

[Cla89] David D. Clark, Van Jacobson, John Romkey, Howard Salwen. "An analysis of TCP processing overhead", *IEEE Communications Magazine*, June 1989, pp. 23-29.

[Cla87a] David Clark, Mark Lambert, Lixia Zhang. NETBLT: A Bulk Data Transfer Protocol. Network Information Center, *RFC-998*, SRI International, March, 1987.

[Cla87b] D. Clark, M. Lambert, L. Zhang. "NETBLT: a high throughput transport protocol", *CCR*, Volume 17, Number 5, 1987, pp. 353-359.

[Col85] R. Colella, R. Aronoff, K. Mills. "Performance Improvements for ISO Transport", *Computer Communication Review,* Vol. 15, No. 5, September 1985.

[Coo90] E. C. Cooper, P. A. Steenkiste, R. A. Sansom, and B. D. Zill. "Protocol Implementation on the Nectar Communication Processor", In *Proceedings ACM SIGCOMM '90,* Philadelphia, PA, September 1990, pp. 135-144.

[Dab94a] Walid S. Dabbous. "High performance presentation and transport mechanisms for integrated communication subsystems", In *Proceedings of the 4th Internatinal IFIP Workshop on Protocols for High Speed Networks*, Vancouver, August 1994.

[Dab93] Walid Dabbous, Christian Huitema. *"XTP implementation under Unix"*, Research Report No 2102, Institut National de Recherche en Informatique et en Automatique, November 1993.

[Dab92] Walid Dabbous et al. *"Applicability of the session and the presentation layers for the support of high speed applications"*, Technical Report No 144, Institut National de Recherche en Informatique et en Automatique, October 1992.

[Dab91] W. Dabbous, *"Etude des protocoles de contrôle de transmission à haut débit pour les applications multimédias"*, PhD Thesis, Université de Paris-Sud, March 1991.

[Diaz94] M. Diaz, C. Chassot, and A. Lozes. "From the Partial Order Connection Concept to Partial Order Multimedia Connections". First HIPPARCH workshop, INRIA Sophia Antipolis, December 15-16, 1994.

[Diot92] Christophe Diot, Patrick Coquet and Didier Stunault. *"Specifications of ETS the Enhanced Transport Service"*. Research Report 907-I, LGI-Institut IMAG, May 1992.

[Diot95] C. Diot, I. Chrisment, A. Richards. "Application Level Framing and Automated Implementation", 6th IFIP International Conference on High Performance networking, Palma (Spain), September 1995.

[Dru93] Peter Druschel, Mark B. Abbott, Michael A. Pagels, and Larry Peterson. "Network Subsystem Design", *IEEE network*, July 1993, pp. 8-17.

[Fel93] D. C. Feldmeier. "A Survey of High Performance Protocol Implementation Techniques", *High Performance Networks - Technology and Protocols*, Ed. Ahmad Tantawy, Kluwer Academic Publishers. Boston, MA, 1993, pp. 29-50.

[Gun91] P. Gunningberg, C. Partridge, T. Sirotkin, B. Victor. "Delayed evaluation of gigabit protocols", In *Proceedings of the* 2^{nd} *MultiG Workshop*, 1991.

[Hui91] Christian Huitema. *"MAVROS, Highlights on an ASN.1 compiler"*, INRIA research note, May 1991.

[Hui90] Christian Huitema. *"Definition of the Flat Tree Light Weight Syntax (FTLWS)"*, Internal Document, INRIA Sophia Antipolis, July 1990.

[Hui89] Christian Huitema, Assem Doghri. "Defining faster transfer syntaxes for the OSI Presentation Protocol", *Computer Communication Review*, Vol 19, No 5, Oct 1989, pp. 44-55.

[Hogl94] Anna Hoglander. *Experimental evaluation of TCP in user space.* INRIA internal report, request from `cdiot@sophia.inria.fr`.

[IEEE83] IEEE Special Issue on Open Systems Interconnection (OSI). Proc. IEEE, vol. 71, no. 12, December 1983, pp 1329-1488.

[Kan88] Hemant Kanakia, David R. Cheriton. "The VMP Network Adapter Board (NAB): High-Performance Network Communication for Multiprocessors", In *Proceedings SIGCOMM '88*, Stanford, CA, 1988, pp. 175-187.

[Lap92] T. F. La Porta and M. Schwartz. "A high-Speed Protocol Parallel Implementation: Design and Analysis", In *Proceedings of the* 4^{th} *IFIP Conference on High Performance Networking*, Liège, Belgium, December 1992.

[Lei85] B.M. Leiner, R.H. Cole, J.B. Postel, D. Mills. "The DARPA Internet Protocol Suite", In *Proceedings INFOCOM'85*, IEEE, March 1985.

[Léo92] L. Léonard. "Enhanced Transport Service Specification". *Deliverable ULg-4*, OSI 95 project, October 1992.

[O'M90] S. W. O'Malley and L. L. Peterson. "A Highly Layered Architecture for High-Speed Networks", In *Proceedings of the IFIP Workshop on Protocols for high speed networks II,* Palo Alto, CA, 1990, pp. 141-156.

[Omal94] S. W. O'Malley, T. Proebsting, and A. B. Montz. "USC : A Universal Stub Compiler". In Proceedings of ACM SIGCOMM'94. Vol. 24, No 4. October 1994.

[Oech94] P. Oechslin, S. Leue. "Enhancing Integrated Layer Processing using Common Case Anticipation and Data Dependence Analysis", In Proceedings of the 1st International Workshop on High Performance Protocol Architectures, December 15-16, 1994, Sophia-Antipolis, France

[Par93b] C. Partridge and S. Pink. A Faster UDP. Submitted to *IEEE Transaction on Networking.*

[Peh92] Bjorn Pehrson, Per Gunningberg and Stephen Pink "Distributed Multimedia Applications on Gigabit Networks", *IEEE Network Magazine*, Vol 6, No 1, January 1992, pp. 26-35.

[Plag94] T. Plagemann, B. Plattner, M. Vogt, T. Walter. "A Model for Dynamic Configuration of Light-Weight Protocols", In *Proceedings of the third workshop on FTDCS*, Tapei. Taiwan. pp. 100-110. April 1992.

[Rich94] A. Richards, A. Seneviratne, M. Fry and V. Witana. "Tailoring the Transport Protocol for Giga Bit Networks". In *Proceedings of the Australian Telecommunication Networks and Applications Conference.* 5-7 December 1994.

[Rüt92] Erich Rütsche and Matthias Kaiserwerth. "TCP/IP on the Parallel Protocol Engine", In *Proceedings of the* 4^{th} *IFIP Conference on High Performance Networking*, Liège, Belgium, December 1992.

[Sch93] D. Schmidt, B. Stiller, T. Suda, A.N. Tantawy, and M. Zitterbart. "Language Support for Flexible Application-Tailored Protocol Configuration", *Proceedings of LCN '93.*

[Shen94] S. Shenker. "Fundamental Design Issues for the Future Internet", preprint submitted to JSAC. 1994.

[Ten89] David L. Tennenhouse. "Layered Multiplexing considered harmful", In *Proceedings of the IFIP Workshop on Protocols for high speed networks,* Zurich, Switzerland, 9-11 May, 1989.

[Turl94a] T. Turletti, C. Huitema. "Packetization of H.261 video streams", Internet Draft, Sept. 1994.

[Turl94b] T. Turletti. "The INRIA Videoconferencing System (IVS)", ConneXions - The Interoperability Report, Vol. 8, No 10, October 1994, pp. 20-24.

[Wak92] Ian Wakeman, Jon Crowcroft, Zheng Wang, and Dejan Sirovica, "Layering considered harmful", *IEEE Network*, January 1992, p. 7-16.

[Wat87] Richard W. Watson, Sandy A. Mamrak. "Gaining efficiency in transport services by appropriate design and implementation choices", *ACM transactions on computer systems,* Vol 5, No. 2, May 1987, pp 97-120.